N94-18375
11.3.1

3rd NASA Symposium on VLSI Design 1991

Canonical Multi-Valued Input Reed-Muller Trees and Forms

M. A. Perkowski [1] and P. D. Johnson
Department of Electrical Engineering
Portland State University
P.O. Box 751, Portland, Oregon 97207

Abstract - **There is recently an increased interest in logic synthesis using EXOR gates. The paper introduces the fundamental concept of *Orthogonal Expansion*, which generalizes the ring form of the Shannon expansion to the logic with multiple-valued (mv) inputs. Based on this concept we are able to define a family of canonical tree circuits. Such circuits can be considered for binary and multiple-valued input cases. They can be multi-level (trees and DAGs) or flattened to two-level AND-EXOR circuits. Input decoders similar to those used in Sum of Products (SOP) PLAs are used in realizations of multiple-valued input functions. In the case of the binary logic the family of flattened AND-EXOR circuits includes several forms discussed by Davio and Green. For the case of the logic with multiple-valued inputs, the family of the flattened mv AND-EXOR circuits includes three expansions known from literature and two new expansions.**

1 Introduction

Although the EXOR gate exists in most VLSI cell libraries, there are no logic synthesis systems that find optimized multi-level circuits using EXORs. The recently developed PLD devices, such as Programmable Gate Arrays (Xilinx LCA 3000) [33], Signetics LHS501 [32], Actel [7] or other [13], either include EXOR gates, or allow to realize them in the "universal logic modules". Since the five input EXOR gate in Xilinx device has the same speed and cost as, for instance, a five input OR gate [5], the new design methods are neeeded for such technologies that will assume the usage of EXOR gates on the same full rights as the AND and OR gates. Particularly, if a Reed-Muller [15,22] form has less terms than a two-level AND-OR expression, this form should be used for Xilinx realization, and not the SOP expression, as it is done nowadays.

The problem of finding the minimal generalized Reed-Muller (GRM) canonical form of optimal polarity [14] (called also fixed-polarity Reed-Muller [9]), as well as the problem of finding the minimal Exclusive Sum of Products (ESOP) of a Boolean function [2,10,27,28], are the classical ones in logic synthesis theory, but exact solutions to them have been proposed for only small functions [16,17].

Solving the above two problems, and creating other new methods of multi-level EXOR circuits design is practically important for several reasons: (1) It has long been the experience of logic designers, that the EXOR circuits can be more economical than the

[1]This research was supported in part by the NSF Research Initiation Award for the first author

conventional inclusive (AND-OR) normal circuits. This was also confirmed practically on many practical examples, especially on arithmetic and telecommunication circuits [1,12]. It was also proven theoretically [26] on worst case and arithmetic functions. (2) The structure of EXOR circuits implementations is especially suitable for VLSI, optical, and some other recent technologies. The RM and GRM forms have absolutely superior design-for-test properties [6,11,21,23], unmatched by other realizations. This was not used in the past since the EXOR gate realizations were slow and area-expensive. With the arrival of PGA devices this deficiency no longer holds and the theories developed for instance in [6,11,21,23] should be practically used. (3) Currently, the widely used logic minimization programs such as Espresso [3] and MIS II do not take into account EXOR gates in their minimization processes which often causes nonminimal results. There is a growing industrial interest among CAD logic synthesis tools users community to have a program that would generate optimized circuits including EXOR gates [12], and such tools start to be introduced to CAE market (for instance by Mentor Graphics Inc.). (4) The new tools for ESOP synthesis are either heuristic [2,10,18,27,28] or produce exact solutions for general ESOPs [16,17,20], but are so slow that can be applied only to small functions. For few canonical forms included in ESOPs optimal programs exist for functions of about 10 variables [29,30,31]. It is therefore important to construct programs that will be faster than the current exact minimizers and still be able to produce quasi-minimal solutions.

A book by Davio [4] and a paper by Green [9] give information on the numbers and properties of various canonical forms being specializations of binary ESOPs, which may be useful to create efficient algorithms for them. In [19] we presented a family of *multiple-valued input expansions*. In this paper we will present a subset of the family from [19], but we will present the material in a more complete and systematic way. We will introduce new canonical binary and multiple-valued forms and expressions. Forms, Directed Acyclic Graphs (DAGs), Trees and expressions obtained by the introduced here tree searching methods will be all called *expansions*. The ultimate goal of the research reported here is to create synthesis programs, exact and approximate, for all known and some new forms being subsets of binary and multiple-valued input ESOPs.

2 Binary Generalizations of Reed-Muller Forms

A Reed-Muller (RM) expression (for binary logic) is an exclusive sum of products of *positive* (non-complemented) input variables. *A Negative Reed-Muller (NRM) expression* is an exclusive sum of products of *negative* (complemented) input variables. Both these expansions are called *Single Polarity Reed-Muller Forms.*

Definition 2.1. The *literal* x_i^c is a variable x_i in either positive (x_i) or complemented ($\bar{x}_i$) form.

Let us consider the following form:

$$f(x_1, ..., x_n) = g_0 \oplus g_1 x_1^c \oplus .... \oplus g_n x_n^c \oplus g_{n+1} x_1^c x_2^c \oplus .... \oplus g_{2^n-1} x_1^c x_2^c ... x_n^c \quad (1)$$

where: $g_i = 0$ *or* 1, and $x_i^c = x_i$ *or* $\bar{x}_i$.

Definition 2.2. By a *Generalized Reed-Muller Form (GRM)* one understands a form 1 in which each variable can be complemented (negative) or not complemented (positive), but can not stand in both forms.

Such forms are canonical, which means that only one such form exists for every polarity of variables (there are 2^n such polarities for a Boolean function of n binary inputs, which means that there are 2^n corresponding GRM forms). Applying the principle of duality to all presented forms one gets the dual forms: the system ($\oplus$, $\cdot$) is replaced with the dual system ($\odot$, $+$). All results of this paper, after applying the principle of duality to them, hold in the dual system as well. Let us observe that the circuits generated for both systems can be implemented using EXOR and NOR gates or EXOR and NAND gates.

By "flattening" we understand applying the Boolean rule, $a(b \oplus c) = a\,b \oplus a\,c$. Flattening is used to convert trees and multi-level expressions to two-level expressions, such as Reed-Muller forms, or ESOPs.

The well-known Shannon expansion for the case of ESOP expansion is as follows:

$f(x_1, ..., x_i, ..., x_n) =$

$$\bar{x}_i \cdot f(x_1, ..., x_i = 0, ..., x_n) \oplus x_i \cdot f(x_1, ..., x_i = 1, ..., x_n) \tag{2}$$

By applying laws $\bar{a} = 1 \oplus a$ and $a = 1 \oplus \bar{a}$ one gets:

$f(x_1, ..., x_i, ..., x_n) =$

$$f(x_1, ..., x_i = 0, ..., x_n) \oplus x_i \cdot [f(x_1, ..., x_i = 0, ..., x_n) \oplus f(x_1, ..., x_i = 1, ..., x_n)] \tag{3}$$

and

$f(x_1, ..., x_i, ..., x_n) =$

$$f(x_1, ..., x_i = 1, ..., x_n) \oplus \bar{x}_i \cdot [f(x_1, ..., x_i = 0, ..., x_n) \oplus f(x_1, ..., x_i = 1, ..., x_n)] \tag{4}$$

In the short form:

$$f = x_i \cdot f_{x_i} \oplus \bar{x}_i \cdot f_{\bar{x}_i} \tag{5}$$

$$f = f_{\bar{x}_i} \oplus x_i \cdot [f_{x_i} \oplus f_{\bar{x}_i}] \tag{6}$$

$$f = f_{x_i} \oplus \bar{x}_i \cdot [f_{x_i} \oplus f_{\bar{x}_i}] \tag{7}$$

Let us observe that these expansion formulas have been applied by several authors for the synthesis of GRM forms for completely specified functions [4]. Davio [4] and Green [9] use them as a base of *Kronecker Reed-Muller (KRM)*, *Pseudo-Kronecker Reed-Muller (PKRM)*, and *Quasi-Kronecker Reed-Muller (QKRM)* forms (Green uses also trees for better explanation). If only rule 6 is used repeatedly for some fixed order of expansion variables, the RM Trees are created, which correspond to RM forms after their flattening. If for every variable one uses either rule 6 or rule 7, the GRM Trees are created, from which GRMs are obtained by flattening (which proves in other way why there is 2^n of such forms). If for every variable one uses either rule 5, rule 6, or rule 7, the KRM Trees are created, from which KRMs are obtained by flattening (which proves in other way why there is 3^n of such forms). If rules 5, 6 and 7 are used, but in each subtree there is a choice of a

rule, the PKRM Trees are generated from which PKRM forms are obtained by flattening. Now, if additionally we allow the expansion variables to have various orders (but the same in the entire tree), one obtains the QKRM Trees, and PKRM flattened forms, respectively. One can now see that a further natural generalization is to allow various orders of variables in subtrees of QKRM trees to create an even wider family of trees. There are two ways of generalizing those forms for the logic with multiple-valued inputs. One was shown in [19]. The other one will be presented here.

3 Generalizations of Reed-Muller Forms for the Logic with Multiple-Valued Inputs

Definition 3.1. A multiple-valued input, two-valued output, completely specified switching function f *(multiple-valued function,* for short) is a mapping: $f(X_1, X_2, \ldots, X_n)$: $P_1 \times P_2 \times \ldots P_n \rightarrow \{0,1\}$, where X_i is a *multiple-valued variable,* $P_i = \{0, 1, \ldots, p_i - 1\}$ is a *set of truth values* that this variable may assume. This is a generalization of an ordinary n-input switching function f: $\{0,1\}^n \rightarrow \{0,1\}$.

Definition 3.2. For any subset $S_i \subseteq P_i$, ${X_i}^{S_i}$ is a *literal* of X_i. The set of values S_i will be called the *polarity of literal* ${X_i}^{S_i}$. The literal ${X_i}^{S_i}$, where $S_i \in P_i$ is defined as follows: ${X_i}^{S_i} = 1$ if $X_i \in S_i$; and ${X_i}^{S_i} = 0$ otherwise.

Example 3.1. For values 0,1 or 2 of a 5-valued variable X, the literal $X^{0,1,2}$ equals 1. For values 3 or 4 of a 5-valued variable X, the value of the literal $X^{0,1,2}$ equals 0.

Definition 3.3. A product of literals, ${X_1}^{S_1}\ {X_2}^{S_2} \ldots {X_n}^{S_n}$, is referred to as a *product term* (also called *term* or *product* for short). A sum of products is denoted as a (multi-valued input) *sum-of-products expression (SOPE).*

Example 3.2. 2-bit decoders have pairs of primary inputs of the function as their inputs. Assume pairing of variables $X_1 = (x_i, x_j)$. The corresponding 2-bit decoder has two input variables; x_i and x_j, and $2^2 = 4$ outputs: $\bar{x}_i \cdot \bar{x}_j$, $\bar{x}_i \cdot x_j$, $x_i \cdot \bar{x}_j$, $x_i \cdot x_j$. Those outputs correspond to the following literals of variable X_1 : ${X_1}^{0,1,2}$, ${X_1}^{0,1,3}$, ${X_1}^{0,2,3}$, ${X_1}^{1,2,3}$, respectively.

Switching functions with multiple-valued inputs, two-valued outputs, find several applications in logic design, pattern recognition, and other areas. In logic design, they are primarily used for the minimization of PLAs that have 2-bit decoders on the inputs. A Programmable Logic Array (PLA) with r-bit input decoders directly realizes a SOPE of a 2^r-valued input, two-valued output, function [25]. Such decoders can be also used in any other realization of the logic with multiple-valued inputs, like multiple-valued input ESOPs [18,27]. A simplified form of such decoders was used in [29,30,31] in the realization of *Multiple-Valued Input Kronecker Reed-Muller Forms (MIKRMs).* It will be also used in the "fixed-polarity" *Multi-Valued Input Kronecker Reed-Muller Trees (MIKRMTs)* that will be introduced here. This simplification consists in creating a simplified decoder with $2^r - 1$ outputs for r input signals. The set of outputs of the simplified decoder is a subset of functions (all but one) realized by a standard decoder. For instance, for a 4-valued input signal X one needs any three outputs of a standard decoder from Example 3.2.

In the case of binary-input logic, each variable x_i from a GRM form can have one of two possible *polarities,* 0 or 1. The notation used for binary functions is: $x_i^0 = \bar{x}_i$, $x_i^1 = x_i$. Let us observe that if two polarities were available for even a single variable, then the ESOP expression including literals of both polarities would be not canonical, for instance x and $1 \oplus \bar{x}$ would represent two different expressions for the same function $f(x) = x$.

The question arises, how to create canonical generalized Reed-Muller forms for multiple-valued input logic. Methods were shown in [19,29,30,31]. Here we will present another method, that allows for more general interpretations. It is next used to create the family of canonical forms and trees. Let us first observe that in a logic with a p_i-valued input X_i there exists p_i different single logic values: for variable X_i one can create p_i different literals with arbitrary single-value polarities. It is obvious that if we will take all those literals to the ESOP expression, then there will be more that one way to describe any Boolean variable function of a single variable. If a single literal from this set of literals is removed, then the remaining literals describe any single-input function in an univocal (canonical) way. For instance, for a p-valued variable X one has the literals: X^{i_0}, X^{i_1}, X^{i_2}, ... , $X^{i_{p-1}}$. Removing any one of them, say X^2, one gets the following literals: X^0, X^1 , X^3 , X^4 ,..., X^{p-1}. Such literals will be called *allowed literals.* Literal X^2 is univocally created as $1 \oplus X^0 \oplus X^1 \oplus X^3 \oplus ... \oplus X^{p-1}$. It can be proven [29,30,31] that for a GRM expansion one can take any p-1 single-valued literals, and moreover, any p-1 literals that form an orthogonal *polarity matrix.* For instance for p=4 one can have the following set of allowed literals: $\{X^{0,1,2}, X^{0,1,3}, X^{0,2,3}\}$, which is described by a polarity matrix:

$$PM(X) = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} X^{0,1,2} \\ X^{0,1,3} \\ X^{0,2,3} \end{bmatrix} \tag{8}$$

It is assumed that logic value **1** (universe) is available. This corresponds to a literal with all possible values, which in turn means a row of all ones in the "expanded" polarity matrix.

The orthogonal expanded polarity matrix includes also a row of ones which corresponds to the universe **1**. For the above example the expanded polarity matrix is:

$$EPM(X) = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} X^{0,1,2} \\ X^{0,1,3} \\ X^{0,2,3} \\ X^{0,1,2,3} \end{bmatrix} = \begin{bmatrix} X^{0,1,2} \\ X^{0,1,3} \\ X^{0,2,3} \\ \mathbf{1} \end{bmatrix} \tag{9}$$

Let us observe that all possible literals can be created by exoring rows of this matrix. *The Expanded Polarity Matrix* of variable X_i is also called *polarity* of this variable. Let us observe that there are the following expanded polarity matrices of binary variables:

POLARITY-2-1-A: $EPM(X) = \begin{bmatrix} 1 \\ X^0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$

POLARITY-2-1-B: $EPM(X) = \begin{bmatrix} 1 \\ X^1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$

POLARITY-2-2: $EPM(X) = \begin{bmatrix} X^0 \\ X^1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Let us observe that when all variables are in polarity POLARITY-2-1-B the function is in the binary Reed-Muller form. When all variables are in polarity POLARITY-2-1-A the function is in the binary Negative Reed-Muller form. When all variables are in polarity POLARITY-2-2 the function is in the binary canonical AND-EXOR minterm form. When each variable is either in polarity POLARITY-2-1-A or in polarity POLARITY-2-1-B, the function is in a GRM form. Applying expansions of variables according to POLARITY-2-1-A or polarity POLARITY-2-2, we obtain a new (mixed polarity) canonical form. Similarly, applying expansions of variables according to POLARITY-2-1-B or polarity POLARITY-2-2, we obtain another new (mixed polarity) canonical form. Applying expansions of variables according to POLARITY-2-1-A, POLARITY-2-1-B or polarity POLARITY-2-2, we obtain the Kronecker Reed-Muller canonical form [4,8]. The concept of the polarity matrix will allow now to generalize the concept of canonical trees and forms to the logic with multiple-valued inputs.

Any form in which all variables are in the same polarity is called a Multiple-Valued Input, Binary Output Restricted GRMs (MIRGRM) form. Such form is canonical since the expansion is unique for each of its variables. It can be shown that for a logic with 3-valued inputs there are 29 various polarities, and 29 MIRGRMs. The number of MIRGRM forms for a logic with p-valued inputs can be calculated from the known mathematical results on the number of orthogonal zero-one matrices. Assuming that universe **1** is available (which is reasonable for practical reasons), expansions that use row of ones in expanded polarity matrix are more interesting. Under such assumption, some examples of sets of allowed literals for a 4-valued input variable X are: $\{X^{0,1,2}, X^{0,1,3}, X^{1,2,3}\}$, $\{X^{1,3}, X^{2,3}, X^{3}\}$, $\{X^{0,2}, X^{0,1}, X^{0,1,2}\}$, $\{X^{1,3}, X^{2,3}, X^{1,2,3}\}$. It can be easily checked that for all those sets a complete set of all literal values can be obtained from other allowed literals by exoring rows of the expanded polarity matrix. There are examples of using switching functions with such literals for practical circuits such as adders [29,30,31].

Reed-Muller forms are extremely easily testable [6,21,23]. We have proved in a forthcoming paper that also all the generalized binary (and even multiple-valued) Reed-Muller forms discussed here have very good testability properties. Among MIRGRMs for 4-valued logic especially preferable is the form which corresponds to the set of allowed literals: $\{X^{1,3}, X^{2,3}, X^{1,2,3}\}$, since the decoder is very simple - a single OR gate: $x_1 = X^{1,3}$, $x_2 = X^{2,3}$, $x_1 + x_2 = X^{1,2,3}$. The test generation for this form is easy (it uses an adaptation of methods from the literature). It minimizes the total layout area comparing to other decoders, because of small area of the OR gate.

Definition 3.4. The set of *allowed literals* for a p-valued variable X is a set with $p-1$ elements whose corresponding polarity matrix is orthogonal.

Definition 3.5. Allowed literal is a literal with the set of values corresponding to a row of an orthogonal polarity matrix.

Definition 3.6. Polarity vector PV $= [PM_1, PM_2, ..., PM_n]$ is a vector of polarity matrices of input variables.

Definition 3.7. By a *Multiple-Valued Input Kronecker Reed-Muller (MIKRM) Expression* for a polarity vector one understands an exclusive sum of products in which all (multiple-valued) literals are allowed literals for this polarity vector.

It can be proven [31] that MIKRM expression is canonical (which means that if for each variable a single polarity is selected, then there exists only one MIKRM expression for this set of variables and their corresponding polarities). It results directly from the fact that for each of its input variables there exists a unique expansion. Therefore we will refer from now on to a MIKRM form, remembering that there are many such forms for a function. Since for ternary variables there are 29 polarities, there are 29^n MIKRMs for a function of n ternary variables. If the universe is not a row of an EPM(X) then all terms of MIKRM need to include literals of variable X. This is of course of only theoretical interest, since in the existing technologies the universe (logic constant **1**) is available at no cost.

As we can see, the MIKRM form is a generalization of the concepts of GRM and KRM forms. It can be observed that there are no separate generalizations of GRMs and KRMs for the logic with multiple-valued inputs.

It results from the above definitions that the MIKRM class is properly included in the ESOP class. The introduced above concepts and definitions will be now illustrated with an example.

Example 3.3. Assuming 4-valued input variables X and Y, the expression:

$$f(X,Y) = 1 \oplus X^{0,1,2}\, Y^{0,1,2} \oplus X^{0,1,3}\, Y^{2,3} \oplus X^{1,2,3}\, Y^{2,3} \oplus X^{0,2,3}\, Y^{1,2,3}$$

is an ESOP but it is not a MIKRM form because there exists the variable X that has four different polarities, while only three polarities are allowed for it. The equivalent MIKRM can be obtained by the replacement of the fourth literal of variable X by an EXOR combination of its another literals:

$$f(X,Y) = 1 \oplus (1 \oplus X^{0,1,3} \oplus X^{1,2,3} \oplus X^{0,2,3})\, Y^{0,1,2} \oplus X^{0,1,3}\, Y^{2,3} \oplus X^{1,2,3}\, Y^{2,3} \oplus X^{0,2,3}\, Y^{1,2,3}.$$

The rule $X^{0,1,2} = 1 \oplus X^{0,1,3} \oplus X^{1,2,3} \oplus X^{0,2,3}$ can be written as: $1111 \oplus 1101 \oplus 0111 \oplus 1011 = 1110$. By using the "flattening" Boolean rule the expression $f(X,Y)$ can be now converted to the exor of products form:

$$f(X,Y) = 1 \oplus Y^{0,1,2} \oplus X^{0,1,3}\, Y^{0,1,2} \oplus X^{1,2,3}\, Y^{0,1,2} \oplus$$

$$X^{0,2,3}\, Y^{0,1,2} \oplus X^{0,1,3}\, Y^{2,3} \oplus X^{1,2,3}\, Y^{2,3} \oplus X^{0,2,3}\, Y^{1,2,3}.$$

As we can easily verify, this last form is a MIKRM, since all literals are now allowed, and

$$PM(X) = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}, \quad PM(Y) = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

4 The Orthogonal Expansion for Multiple-Valued Input Switching Functions

Let us assume that a Boolean function in a form of ESOP is represented as a list of terms called ARESOP. In particular, it can be a set of minterms, or a set of disjoint cubes

representing both a SOP and an ESOP. Our algorithms, however, can assume *arbitrary* ESOP, for any kind of orthogonal expansion.

Let us assume that given is a *vector of expanded polarity matrices:*

$$PV = [PM_1, PM_2, ..., PM_n].$$

To perform an expansion of an ESOP ARESOP with respect to an expanded polarity matrix PM(X_i) of variable X_i, one has to convert every literal of variable X_i in all cubes from the ARESOP to an EXOR combination of literals that are allowed for this polarity (a universe (a vector of ones) is treated as an allowed literal as well). If a cube CUB of ARESOP has no literal $X_i{}^S$ and the universe is absent from the expanded polarity matrix, cube CUB should be first represented as $\mathbf{1} \cdot CUB$, and next the universe **1** from it should be converted to the EXOR combination of literals allowed for variable X_i. (This is a generalization of a binary rule $a = a\bar{b} \oplus ab$). It results from the orthogonal properties of the expanded polarity matrix that such conversion for variable X_i exists and is unique. Next a one level of flattening is executed and the expression is rearranged to the form with all allowed literals factorized. Below we will illustrate the expansion on an example of a function with ternary variables.

Example 4.1.

1. Given is a list ARESOP of disjoint cubes corresponding to expression:

$$X^{0,1}Y^0 \oplus X^1Y^{1,2} \oplus X^2Y^2.$$

2. One has to find expansion with respect to variable X, with allowed literals 1, $X^{0,1}$, $X^{0,2}$. The result of conversion (substitution) is: $X^{0,1}Y^0 \oplus (1 \oplus X^{0,2})Y^{1,2} \oplus (1 \oplus X^{0,1})Y^2$.

3. After flattening: $X^{0,1}Y^0 \oplus Y^{1,2} \oplus X^{0,2}Y^{1,2} \oplus Y^2 \oplus X^{0,1}Y^2$.

4. After factorizing the allowed literals: $X^{0,1}(Y^0 \oplus Y^2) \oplus X^{0,2}(Y^{1,2}) \oplus 1(Y^{1,2} \oplus Y^2)$.

In the next stage similar expansion is done for variable Y. The expansion uses the respective expanded polarity matrix EPM(Y).

Two computer-oriented efficient algorithms to perform this kind of expansion for flat forms are given in [29,30,31] and illustrated with examples there. They do not use flattening and factorizing, however, they cannot be also applied to create tree expansions.

Below we will introduce the basic concept of EXOR type Shannon expansion for the mv logic. As it is well-known, the Shannon expansion theorem has been generalized by Rudell [24] for the mv logic. His expansion is of AND-OR type, to be used for mv SOP synthesis. On the other hand, three generalizations of Shannon theorem for Boolean rings are known [4,9] (rules 5,6 and 7 in section 2). Here we will formulate an expansion that generalizes both Rudell's and Davio's expansions: it is in terms of AND-EXOR expansion, and it is for mv logic.

It can be derived that the orthogonal expansion of function f with respect to multiple-valued input variable X_i of expanded polarity matrix $EPM(X_i)$ can be expressed by the following formula:

$$f = \bigoplus_{X_i^{S_j} \in EPM(X_i)} f_{X_i^{S_j}} \; X_i^{S_j} \tag{10}$$

where the values of $f_{X_i^{S_j}}$ are calculated as follows: $[f_{X_i^{S_j}}]^T = [f_{X_{i,j}}]^T [NP]^{-1}$; $[f_{X_i^{S_j}}]$ is a vector of single-literal orthogonal expansions of literal $X_i^{S_j}$, $j = 0,...,p-1$; $[f_{X_{i,j}}]$ is a vector of single-literal standard expansions of single-value literal X_i, $(X_i = j)$, $j = 0,...,p-1$; $[A]^T$ means matrix $[A]$ transpose; $[A]^{-1}$ means matrix $[A]$ inverse; $[NP]$ is a normalized polarity matrix, which relates polarities of multiple-valued input literals to single-value literals.

Instead of proving this expansion for a general case we will sketch the proof using another example.

Example 4.2. The expanded polarity matrix for ternary variable X_i is:

$$EPM(X_i) = \begin{bmatrix} X^{0,2} \\ X^{0,1} \\ X^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{11}$$

According to formula 10 the orthogonal expansion for EPM(X_i) is:

$$f = f_{X_i^{0,2}} \; X_i^{0,2} \oplus f_{X_i^{0,1}} \; X_i^{0,1} \oplus f_{X_i^2} \; X_i^2. \tag{12}$$

We will derive the values of $f_{X_i^{0,2}}$, $f_{X_i^{0,1}}$, and $f_{X_i^2}$. It holds for non-overlapping literals [24]: $f = f_{X_{i,0}} \; X_i^0 + f_{X_{i,1}} \; X_i^1 + f_{X_{i,2}} \; X_i^2$ which, with respect to the disjointness of X_i^r, X_i^s, $s \neq r$, gives: $f = f_{X_{i,0}} \; X_i^0 \oplus f_{X_{i,1}} \; X_i^1 \oplus f_{X_{i,2}} \; X_i^2$. Then:

$$f = f_{X_{i,0}} \; X_i^0 \oplus f_{X_{i,1}} \; X_i^1 \oplus f_{X_{i,2}} \; X_i^2 = f_{X_i^{0,2}} \; X_i^{0,2} \oplus f_{X_i^{0,1}} \; X_i^{0,1} \oplus f_{X_i^2} \; X_i^2 \tag{13}$$

$$\begin{aligned} f &= (f_{X_i^{0,2}} \; X_i^0 \oplus f_{X_i^{0,2}} \; X_i^2) \oplus (f_{X_i^{0,1}} \; X_i^0 \oplus f_{X_i^{0,1}} \; X_i^1) \oplus f_{X_i^2} \; X_i^2 = \\ &= (f_{X_i^{0,2}} \oplus f_{X_i^{0,1}}) \; X_i^0 \oplus f_{X_i^{0,1}} \; X_i^1 \oplus (f_{X_i^{0,2}} \oplus f_{X_i^2}) \; X_i^2 \end{aligned} \tag{14}$$

In matrix form, the equation 13 becomes:

$$[f_{X_{i,0}} \;\; f_{X_{i,1}} \;\; f_{X_{i,2}}] \begin{bmatrix} X_i^0 \\ X_i^1 \\ X_i^2 \end{bmatrix} = [f_{X_i^{0,2}} \;\; f_{X_i^{0,1}} \;\; f_{X_i^2}] \begin{bmatrix} X_i^{0,2} \\ X_i^{0,1} \\ X_i^2 \end{bmatrix} \tag{15}$$

The relation between the disjoint and non-disjoint literals is given be the equation:

$$\begin{bmatrix} X_i^{0,2} \\ X_i^{0,1} \\ X_i^2 \end{bmatrix} = [NP] \begin{bmatrix} X_i^0 \\ X_i^1 \\ X_i^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_i^0 \\ X_i^1 \\ X_i^2 \end{bmatrix} \tag{16}$$

Substituting 16 to 15 one obtains:

$$[f_{X_{i,0}} \;\; f_{X_{i,1}} \;\; f_{X_{i,2}}] \begin{bmatrix} X_i^0 \\ X_i^1 \\ X_i^2 \end{bmatrix} = [f_{X_i^{0,2}} \;\; f_{X_i^{0,1}} \;\; f_{X_i^2}] \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_i^0 \\ X_i^1 \\ X_i^2 \end{bmatrix} \tag{17}$$

¿From there:

$$[f_{X_i^{0,2}} \quad f_{X_i^{0,1}} \quad f_{X_i^2}] = [f_{X_{i,0}} \quad f_{X_{i,1}} \quad f_{X_{i,2}}] \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} =$$

$$= [f_{X_{i,0}} \quad f_{X_{i,1}} \quad f_{X_{i,2}}] \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = [f_{X_{i,0}} \oplus f_{X_{i,1}} \quad f_{X_{i,1}} \quad f_{X_{i,0}} \oplus f_{X_{i,1}} \oplus f_{X_{i,2}}] \quad (18)$$

Formula 18 gives the values $f_{X_i^{0,2}}$, $f_{X_i^{0,1}}$, and $f_{X_i^2}$ to be substituted to formula 12 in order to calculate the orthogonal expansion *(End of Example).*

It can be easily checked by substituting respective expanded polarity matrices to formula 10 that the expansions 5 - 7 and the Rudell's expansion are particular cases of this new expansion. This method can be also easily generalized for incompletely specified functions.

1. The orthogonal expansion applied in some restricted way to a multiple-valued input ESOP creates a family of canonical tree expansions analogous to those for binary logic.

2. Applying the expansion uniformly in a tree for a fixed order of expansion variables of the same polarity one obtains the MIRGRM Trees that are the mv counterparts of binary Single Polarity Reed-Muller Trees.

3. Applying the expansion uniformly in a tree for a fixed order of expansion variables of various polarities one obtains the *Multiple-Valued Kronecker Reed-Muller Trees (MIKRM Trees)* that are the mv counterparts of binary GRM Trees and Kronecker Reed-Muller Trees.

4. Applying the expansion in a tree for a fixed order of expansion variables, but having various variable polarities in different sub-expressions (sub-trees) one obtains the *Multiple-Valued Pseudo-Kronecker Reed-Muller Trees (MIPKRM Trees)* that are the mv counterparts of binary Pseudo-Kronecker Reed-Muller Trees.

5. Applying the expansion in a tree for all possible but fixed orders of expansion variables, and having various variable polarities in different sub-expressions (sub-trees) one obtains the *Multiple-Valued Quasi-Kronecker Reed-Muller Trees (MIQKRM Trees)* that are the mv counterparts of binary Quasi-Kronecker Reed-Muller Trees.

6. Applying the expansion in a tree for all possible orders of expansion variables, having various orders in various sub-trees, and having various variable polarities in different sub-expressions (sub-trees) one obtains a new family of canonical trees.

7. The method can be applied with little modification to multi-output functions: it is applied to each function separately. The logically equivalent sub-trees are be combined, which leads to DAG circuits. This transformation preserves the canonicity of the tree circuits.

8. The trees from all above new families of canonical trees can be flattened to respective canonical mv forms. This leads to MIRGRM forms, MIKRM forms, MIPKRM forms, MIQKRM forms, and new mv canonical forms, respectively.

A more detailed characteristics of the above expansions, new mv expansions and computer algorithms to create them will be included in our forthcoming paper.

5 Conclusion

In this paper several well-known canonical forms have been generalized for the logic with multiple-valued inputs. An Orthogonal Expansion Theorem has been also formulated, which plays that fundamental a role in those expansions as one played by the Shannon Theorem in inclusive logic and the three Boolean ring expansions for the binary forms. Since the Shannon theorem has several important application in tautology, complementation, implicants generation and many other areas, and the ring expansions are fundamental to EXOR circuits theories, we expect this theorem to play also a fundamental role in the multiple-valued logic.

The reader must bear in mind that the expansions proposed here relate to trees and not "flat" forms. For instance, the GRM forms are independent on the order of variables, but the respective GRM trees do depend on this order. Therefore, investigating expansions with changing the order of variables has practical sense only for some types of expansions. Since several expansions obtained by changing the order of variables produce the same "flat" form, counting of several forms can be difficult, as already observed for Quasi-Kronecker forms by Green [4]. It is even more so for our forms, where different orders of variables in subtrees are possible.

References

[1] Ph.V. Besslich, "Efficient Computer Method for EXOR Logic Design", *Proc. IEE*, Vol. 130, Part E, CDT, No. 6., pp. 203-206, 1983.

[2] D. Brand, T. Sasao, "On the Minimization of And-Exor Expressions", *23 FTC*, pp. 1 - 9, 1990.

[3] R.K. Brayton, G.D. Hachtel,C.T. McMullen, A.L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, 1984.

[4] P. Davio, J.P. Deschamps, A. Thayse, *Discrete and Switching Functions*. George and McGraw-Hill, New York, 1978.

[5] E. Detjens, "FPGA Devices Require FPGA-specific Synthesis Tools", *Computer Design*, p. 124, Nov 1990.

[6] H. Fujiwara, *Logic Testing and Design for Testability*, Computer System Series, The MIT Press, 1986.

[7] GOTHIC CRELLON, "The Beginners Guide to Programmable ASICs", 1990.

[8] D. Green, *Modern Logic Design*, Electronic Systems Engineering Series, 1986.

[9] D. Green, "Families of Reed-Muller Canonical Forms", *Int. J. Electronics*, Vol. 70, No. 2, pp. 259-280, Jan. 91.

[10] M. Helliwell, M.A. Perkowski, "A Fast Algorithm to Minimize Multi-Output Mixed-Polarity Generalized Reed-Muller Forms", *Proceedings of 25th ACM/IEEE Design Automaton Conference,* 1988, Las Vegas, pp. 427 - 432.

[11] K.L. Kodandapani, "A Note on Easily Testable Realizations for Logical Functions", *IEEE Trans. Comp.*, Vol. C-23, pp. 332-333, 1974.

[12] H. Landmann, "Logic Synthesis at Sun", *IEEE conference paper*, CH 2686 - 4 / 89 / 0000 / 0469, 1989.

[13] MONOLITIC MEMORIES, INC., "XOR PLDs Simplify Design of Counters and Other Devices", *EDN*, May 28, 1987.

[14] A. Mukhophadhyay, G. Schmitz, "Minimization of Exclusive-OR and Logical Equivalence Switching Circuits", *IEEE Trans. Comp.*, Vol. C-19, No. 2., pp. 132-140, February 1970.

[15] D.E. Muller, "Application of Boolean Algebra to Switching Circuit Design and to Error Detection", *IRE Trans. Electron. Comp.*, Vol EC-3, pp. 6-12, September 1954.

[16] G. Papakonstantinou, "Minimization of modulo-2 sum of products", *IEEE Trans. on Computers*, Vol. C-28, pp. 163-167, February 1979.

[17] M.A. Perkowski, M. Chrzanowska-Jeske, "An Exact Algorithm to Minimize Mixed-Radix Exclusive Sums of Products for Incompletely Specified Boolean Functions", *Proc. International Symposium on Circuits and Systems,* May 1990.

[18] M.A. Perkowski, M. Helliwell, P. Wu, "Minimization of Multiple-Valued Input Multi-Output Mixed-Radix Exclusive Sums of Products for Incompletely Specified Boolean Functions", *Proc. IEEE Inter. Symp. Multiple Valued Logic*, Guangzhou, People's Republic of China, May 1989, pp. 256-263.

[19] M.A. Perkowski, P. Dysko, B.J. Falkowski, "Two Learning Methods for a Tree-Search Combinatorial Optimizer", *Proceedings of IEEE International Phoenix Conference on Computers and Communication*, Scottsdale, Arizona, March 1990.

[20] M.A. Perkowski, M. Chrzanowska-Jeske, "Tree Search Algorithms to Find Exact ESOP Forms", *PSU Report*, 1991.

[21] D.K. Pradhan, *Fault-Tolerant Computing. Theory and Techniques. Vol. I.*, Prentice-Hall, 1987.

[22] I.S. Reed, "A Class of Multiple-Error-Correcting Codes and Their Decoding Scheme", *IRE Trans. Inf.Th.*, Vol. PGIT-4, pp. 38-49, 1954.

[23] S.M. Reddy, "Easy testable realizations for logic functions", *IEEE Trans. Comput.*, Vol. C-21, pp. 1183 - 1188, 1972.

[24] R. Rudell, "Multiple-Valued Logic Minimization for PLA Synthesis", Master Thesis, *University of California*, Berkeley, June 1986.

[25] T. Sasao, H. Terada, "Multiple-Valued Logic and the Design of Programmable Logic Arrays with Decoders", *Proc. of 9th International Symposium on Multiple-Valued Logic*, Bath, England, pp. 27 - 37, 1979.

[26] T. Sasao, P. Besslich, "On the Complexity of MOD-2 Sum PLA", *Institute of Electronics and Communication Engineers of Japan*, FTS86-17, pp. 1-8, Nov. 17, 1986.

[27] T. Sasao, "EXMIN: A Simplification Algorithm for Exclusive-OR-Sum-of-Products Expressions for Multiple-Valued Input Two-Valued Output Functions", *Proc. of 20th Int. Symp. on Multiple-Valued Logic*, pp. 128-135, May 1990.

[28] J.M. Saul, "An Improved Algorithm for the Minimization of Mixed Polarity Reed-Muller Representations", *Proc. ICCD'90*, pp. 372-375, Sept. 1990.

[29] I. Schaefer, "An Effective Cube Comparison Method for Discrete Spectral Transformations of Logic Functions", *M. Sc., Thesis*, May 1990.

[30] I. Schaefer, M.A. Perkowski, "Multiple-Valued Input Generalized Reed-Muller Forms", *Proc. of the ISMVL-91*, Victoria, B.C., Canada, May 1991.

[31] I. Schaefer, M.A. Perkowski, "Multiple-Valued Input Generalized Reed-Muller Forms", *submitted to IEE Journal*, May 1991.

[32] SIGNETICS, PLD Data Manual, Signetics' Approach to Logic Flexibility for the '80's", 1986.

[33] XILINX, Inc., *"The Programmable Gate Array Data Book"*, 1989.